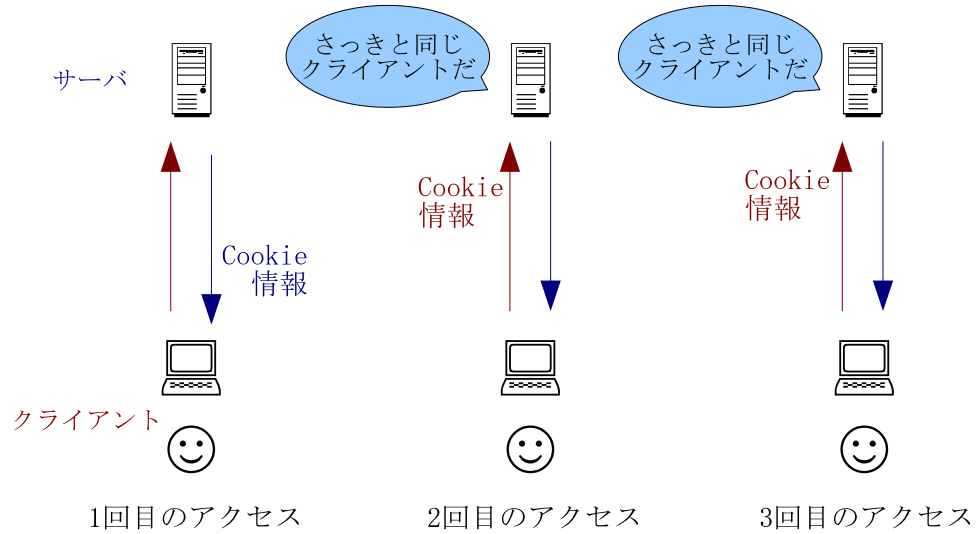
第3章　セッションの利用

3.1　セッションとは

Web サーバーと Web ブラウザーの間の通信 (HTTP による通信) は本来一回ページを表示するごとに完結するものである。つまり、ユーザーが過去にどのようなページを見ていたか、などといった履歴には関係なく動作する。

そこで、過去の履歴（例えば会員制ページのユーザーのログイン情報や、ショッピングサイトの商品の選択（いわゆるショッピングカート）の情報）に依存するような Web アプリケーションを実現するためには、なんらかの形で過去のユーザーのアクセスの情報を保持する必要がある。このようなデータは、Web サイトにアクセスするユーザーごとに管理しなければならないので、単純にフィールドやファイルなどに保存することはできない。このためサーバー側にユーザーごとにデータを保持し、ブラウザーからのページ要求のたびに、何らかの方法でユーザーを識別するためのデータを送信してもらう方法を取る。

このユーザーを識別するデータを送信する方法としては、クッキー（cookie）という技術を使う方法、フォームの隠し要素（hidden）を使う方法, URL 中の Query String に履歴データを埋め込む方法などがある。いずれにしても、店舗の “会員証”・医院の “診察券” に相当するものをブラウザーに渡して、来店・来院する（つまり、ページにアクセスする）たびに提示してもらうようなものである。



問 3.1.1 クッキー（cookie）とは、どういう仕組みか、調べよ。

Java Servlet ではユーザーごとのデータを扱うために**セッション（session）**[1] という仕組みを提供している。セッションは裏側ではクッキーなどの仕組みを使っているが、複雑な部分をプログラマーが見なくても済むようにして、Servlet のプログラマーが簡単にユーザーの履歴データを利用できるインタフェースを提供している。使い方は簡単で、HttpServletRequest クラスの getSession というメソッドでセッションオブジェクト（店舗の “顧客情報”・医院の “カルテ” に相当する）を取り出し、そのオブジェクトに対して、データを関連づけ（setAttribute）たり、読み出し（getAttribute）たりするだけである。クッキーの発行や確認は Web アプケーションサーバーが舞台裏で行っているので、プログラマーが行う必要はない。セッションはユーザーごとに用意されるので、一連のアクセスで以前にセットした値を利用することができる。一方、他のユーザーがセットしたデータは見えない。

3.2　セッションを利用したアクセスカウンター

まず、セッションを利用したアクセスカウンターを紹介する。一見、ファイルを利用したアクセスカウンターとそれほど違いはないが、ユーザーごとにカウンターのデータを保持する。これは異なる Internet Explorer や Firefox など別のブラウザーでアクセスすると（Servlet からは別のユーザーに見えるので）振舞いの違いを確認することができる。

ファイル SessionCounter.java

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3
4  import javax.servlet.ServletException;
5  import javax.servlet.annotation.WebServlet;
6  import javax.servlet.http.HttpServlet;
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletResponse;
9  import javax.servlet.http.HttpSession;
10
11 @WebServlet("/SessionCounter")
12 public class SessionCounter extends HttpServlet {
13   private static final String COUNTER = "counter";
14
15   @Override
16   protected void doGet(HttpServletRequest request,
17                        HttpServletResponse response)
18                 throws ServletException, IOException {
19     response.setContentType("text/html;␣charset=UTF-8");
20     HttpSession session = request.getSession(true);
21     int i = 0;
22     try {
```

[1] もともとは会期などという意味

```
23        i = (int) session.getAttribute(COUNTER);
24      } catch (NullPointerException | NumberFormatException e) {
25         /* i = 0 のまま */
26      }
27      PrintWriter out = response.getWriter();
28      out.println("<html><head></head><body>");
29      out.printf("あなたの␣%d␣回目の御訪問です。", ++i);
30      out.println("</body></html>");
31      session.setAttribute(COUNTER, i);
32      out.close();    // close を忘れない
33    }
34  }
```

HttpServletRequest クラスの getSession メソッドで現在のセッションオブジェクト（HttpSession クラスのオブジェクト）を得る。引数の true はセッションオブジェクトがなければ作成することを示す。

Session クラスの getAttribute メソッドでその値を取り出す。getAttribute メソッドの引数は取り出したいデータに対応づけられた名前で、戻り値がセッションに保存されていたその名前のデータである。getAttribute メソッドの戻り値型は Object 型（つまりすべてのクラスのスーパークラス）なので、String 型や int 型などに型変換（ダウンキャスト・ナローイング）する必要がある。Object 型から int への型変換は int 型のラッパークラスの Integer 型を経由して行なわれる。

最初のアクセスではセッションにデータが保存されていないので、例外が発生し、i の値は 0 になる。

最後に setAttribute メソッドを用いて、セッションオブジェクトにデータを保存する。setAttribute メソッドの第 1 引数は String 型であり、保存するデータに対応させる名前である。この名前は後で getAttribute でデータを取り出すときに用いる。setAttribute メソッドの第 2 引数は実際に保存するデータである。ここにはどのような型のデータが来ることもあり得るため、setAttribute の第 2 引数の型は、すべてのオブジェクトの型を包含する Object という型になっている。Object はすべてのクラスのスーパークラスである。

int 型から Integer 型への型変換（オートボクシング）と Integer 型から Object 型への型変換（アップキャスト・ワイドニング）は暗黙的に行なわれる。

問 3.2.1 アクセス時の時刻をセッションに保存し、次回のアクセス時に「前回は何月何日何時何分何秒にアクセスしました」（または「初めてか久しぶりのアクセスです」）と表示するサーブレット AccessTime.java を（getLastAccessTime メソッドを使わずに）作成せよ。

3.3 Quiz サーブレット

セッションを利用するもう少し大きな Servlet として Quiz サーブレットを例にあげる。これは過去に正解した問題数などによって、表示を変更する Servlet で

ある。

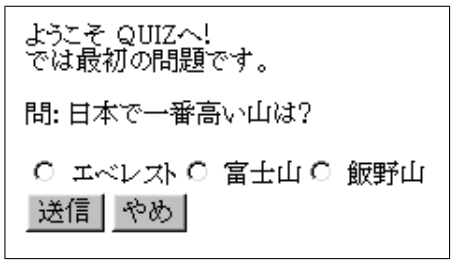
正解した問題数や現在何問めを実行しているかを保存するためにセッションを利用する。（さもないと、複数のユーザーが同時にこのサーブレットにアクセスしたときに、正解した問題数のデータがごっちゃになってしまう。）

例題: QUIZ

ファイル quiz.txt

```
1 日本で一番高い山は? エベレスト 富士山 飯野山 2
2 日本で一番広い湖は? 琵琶湖 府中湖 満濃池 1
3 2020年オリンピック開催予定地は? リオデジャネイロ 東京 北京 2
4 香川大学本部の所在地は? 幸町 花園町 林町 3
```

この Servlet は上のようなテキストファイルから右のような QUIZ を表示するページを作成する Servlet である。



この Servlet では “現在何番目の問題か？” (number) と “これまでの正解数” (score) というデータがセッションのなかに保持されている。(その 1) の部分は特に変わったところはない。ArrayList を使用するのでこれを import している。また、いくつかの定数を定義している。

ファイル Quiz.java（その 1）

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileInputStream;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.io.PrintWriter;
7  import java.util.ArrayList;
8
9  import javax.servlet.ServletException;
10 import javax.servlet.annotation.WebServlet;
11 import javax.servlet.http.HttpServlet;
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14 import javax.servlet.http.HttpSession;
15
16 @WebServlet("/Quiz")
17 public class Quiz extends HttpServlet {
18   private static final String ANSWER    = "answer";
19   private static final String SCORE     = "score";
20   private static final String NUMBER    = "number";
21   private static final String QUESTIONS = "questions";
```

（その 2）の doGet メソッドは最初に Servlet が実行されるときに呼ばれる。このメソッドは単に doPost メソッドを呼び出すだけである。

ファイル Quiz.java（その 2）

```
22  @Override
23  protected void doGet(HttpServletRequest request,
24                       HttpServletResponse response)
25                throws ServletException, IOException {
26    // 最初の問は GET
27    doPost(request, response);
28  }
```

（その 3）は doPost メソッドの最初の部分を示す。ここでは、いくつかの局所変数を宣言していて、セッションオブジェクトを作成している。

ファイル Quiz.java（その 3）

```
29  @Override
30  protected void doPost(HttpServletRequest request,
31                        HttpServletResponse response)
32                throws ServletException, IOException {
33    response.setContentType("text/html;␣charset=UTF-8");
34    PrintWriter out = response.getWriter();
35    out.println("<html><head></head><body>");
36
37    int i, number = 0, score = 0;
38    ArrayList<String[]> questions;
39
40    HttpSession session = request.getSession(true);
```

session.isNew() が真のときは、セッションができたばかりであることを示す。つまり、このページのアクセスが最初の問であることがわかる。

そのときは、QUIZ のデータが入っているファイル（quiz.txt）を開いて、questions という ArrayList に読み込む。この questions の値を次の問以降の表示のときに利用するために、setAttribute メソッドを用いて、セッションオブジェクトに保存する。この例では、ArrayList<String[]>や String などの型から Object 型への型変換が起こっているが、このような上向きの型変換は暗黙に行なわれる。

また、最初の問用のメッセージを表示する。

ファイル Quiz.java（その 4）

```
41    if (session.isNew() || session.getAttribute(QUESTIONS) == null) {
42      // 最初の問
43      questions = new ArrayList<String[]>();
44      File f = new File(getServletContext()
45                .getRealPath("/WEB-INF/quiz.txt"));
46      BufferedReader in = new BufferedReader(new InputStreamReader(
47                            new FileInputStream(f), "UTF-8"));
48      String line = "";
49      while((line = in.readLine()) != null) {
```

```
50          line = line.trim();
51          if(line.trim().equals(""))
52            continue;
53          questions.add(line.split("\\s+")); // 空白の１つ以上の繰返し
54        }
55        in.close();
56        session.setAttribute(QUESTIONS, questions);
57        out.println("<p>ようこそ␣QUIZへ!<br␣/>では最初の問題です。</p>");
58      }
```

一方、session.isNew() が偽のときは、最初の問ではないので、送られたフォームのデータから、 前の問題で解答者が選んだ答の番号（answer）を得る。また、セッションデータには最初の問のときに読み込んだ問題のデータ、現在の問題の番号（"number"）とこれまでの正解数（"score"）が記録されているはずなので、HttpSession クラスの getAttribute メソッドでその値を取り出す。

questions の number-1 番目の最後のトークンを解答と比べる。その結果によってメッセージと score 変数の値を変える。

ファイル Quiz.java（その 5）

```
59        else {
60        // 最初の問ではない
61        try {
62          number = (int)session.getAttribute(NUMBER);
63          score  = (int)session.getAttribute(SCORE);
64          questions = (ArrayList<String[]>)session.getAttribute(QUESTIONS);
65
66          String[] tokens = questions.get(number - 1);
67          int a = Integer.parseInt(tokens[tokens.length - 1]);
68          int answer = Integer.parseInt(request.getParameter(ANSWER));
69          if (a == answer) { // a は最後の文字
70            out.println("正解です。<br␣/>");
71            score++;
72          } else {
73            out.println("残念でした。<br␣/>");
74          }
75        } catch (Exception e) {
76          session.removeAttribute("questions");
77          out.print("想定外のアクセスでエラーが起こりました。");
78          out.println("タブを閉じるかリロードしてください。");
79          e.printStackTrace(out);
80          out.println("</body></html>");
81          out.close();
82          return;
83        }
84      }
```

次に、次の問を表示する準備をする。ただし、number 番めの問題がないときは、クイズを終了する。ファイル Quiz.java（その 6）

```
85      if (number >= questions.size()) { // 終
86        out.println("<br␣/>これで␣QUIZは終わりです。<br␣/>");
87        out.printf("正解数は、%d問でした。%n", score);
88  // session.invalidate();
89        session.removeAttribute("questions");
90      }
```

問題が終わりでなければ、questions から次の問の情報を読み取って、フォームとして出力する。そして各選択肢のラジオボタンと送信ボタン、リセットボタンを出力する。form タグに action 属性がないが、その場合は自分自身（現在表示中のページと同じ URL）にフォームデータを送信する。

最後に setAttribute メソッドを用いて、セッションオブジェクトにこれまでの問題数と正解数を記録している。number を一つ増分してから、setAttribute を呼び出して、number と score をセッションオブジェクトに書き込んでいる。（このデータは次の問題を表示するときに利用される。）

ファイル Quiz.java（その 7）

```
91        else { // 次の問を表示
92        String[] tokens = questions.get(number);
93        out.println("次の問:␣" + tokens[0] + "<br␣/>");
94        out.println("<form␣method='post'>");
95        for (i = 0; i < tokens.length - 2; i++) {
96          out.print("<input␣type='radio'␣name='answer'");
97          out.printf("␣value='%d'  />␣%s", i + 1, tokens[i + 1]);
98        }
99        out.println("<br␣/>");
100       out.println("<input␣type='submit'␣value='送信'␣/>");
101       out.println("<input␣type='reset'␣value='やめ'␣/>");
102       out.println("</form>");
103
104       number++;
105       session.setAttribute("number", number);
106       session.setAttribute("score", score);
107     }
108     out.println("</body></html>");
109     out.close();
110   }
111 }
```

なお、この Servlet を、リロードボタンや戻るボタンなどがクリックされた場合、複数のタブで同時に開いた場合などに、適切に対応するように改良することはなかなか難しい。適切に対処することが必要なときは、生の Servlet ではなく、Wicket などの Web アプリケーションフレームワークを採用することを検討するほうが良いだろう。

問 3.3.1 （選択肢の記録）

Quiz.java を、正解数だけではなくて、各問の選んだ選択肢、正誤までわかるように記録し、その結果を各問の問題文、正答ともに「これで QUIZ は終わりです。」のメッセージのあとにテーブルに整形して表示するサーブレット QuizEx.java を作成せよ。QUIZ の問題数は何問になるかわからないので、問題数に依存しないようにすること。選択肢・正答は番号だけの表示は不可である。（下の表示例を参考にすること。）

これで QUIZ は終了です。

番号	問題文	あなたの答	正答	正誤
問1	日本で一番高い山は?	飯野山	富士山	×
問2	日本で一番広い湖は?	満濃池	琵琶湖	×
問3	2020年オリンピック開催予定地は?	東京	東京	○
問4	香川大学本部の所在地は?	幸町	幸町	○

4 問中、正解は 2 問です。

問 3.3.2　（オプションの選択）
次のようなオプションとその価格が書かれたテキストファイル
ファイル pasocon.txt

```
本体     タワー     20000  スリム    35000  スーパースリム 50000
CPU      Celeron430 4000   DuoE8400 20000  QuadQ9450      37000  ↙
   ↙QuadQ9550 62000
RAM      512MB      2000   1GB       4000  2GB             6000  ↙
   ↙4GB         9000
HDD      80GB       4000   250GB     6000  320GB           6500  ↙
   ↙500GB       8000  1TB  20000
光学D    DVD-R/RW   5000   Blu-ray  20000
```

から次のようなオプション構成を選択できるページを各パーツ毎に作成し、

```
<html>
<body>
CPUを選んで下さい
<br />
<form method='post'>
<input type='radio' name='answer' value='1' />
Celeron430 4000円
<input type='radio' name='answer' value='2' />
DuoE8400 20000円
<input type='radio' name='answer' value='3' />
QuadQ9450 37000円
<input type='radio' name='answer' value='4' />
QuadQ9550 62000円
<br />
```

```
<input type='submit' value='送る' />
<input type='reset' value='キャンセル' />
</form>
</body>
</html>
```

すべてのパーツの選択肢を選んだ時点で、選択した構成の合計価格を表示するサーブレット SelectOptions.java を作成せよ。さらに、見積書のように表の形に整形せよ。

キーワード:

クッキー, hidden（隠し要素）, セッション, HttpSession クラス, getSession メソッド, getAttribute メソッド, setAttribute メソッド, isNew メソッド, invalidate メソッド,